CHOWN

Vulnerable to TOCTOU issues

Sean Barnum, Cigital, Inc. [vita[1]]

2007-03-19

Part "Original Cigital Coding Rule in XML"

Mime-type: text/xml, size: 10528 bytes

Attack Category	• Path spoofing or confusion problem
Vulnerability Category	• TOCTOU - Time of Check, Time of Use • Privilege escalation problem
Software Context	• File Management
Location	• unistd.h
Description	The chown() function sets the owner ID and group ID of the file specified by path or referenced by the open file descriptor fildes to owner and group respectively. If owner or group is specified as -1, chown() does not change the corresponding ID of the file. The lchown() function sets the owner ID and group ID of the named file in the same manner as chown(), unless the named file is a symbolic link. In this case, lchown() changes the ownership of the symbolic link file itself, while chown() changes the ownership of the file or directory to which the symbolic link refers. The fchownat() function sets the owner ID and group ID of the named file in the same manner as chown(). If, however, the path argument is relative, the path is resolved relative to the fildes argument rather than the current working directory. If the fildes argument has the special value FDCWD, the path path resolution reverts back to current working directory relative. If the flag argument is set to SYMLNK, the function behaves like lchown() with respect to symbolic links. If the path argument is absolute, the fildes argument is ignored. If the path argument is a NULL pointer, the function behaves like fchown(). If chown(), lchown(), fchown(), or fchownat() is invoked by a process other than superuser, the set-user-ID and set-group-ID bits of the file mode, S_ISUID and S_ISGID respectively, are cleared.

1. http://buildsecurityin.us-cert.gov/bsi-rules/35-BSI.html (Barnum, Sean)

chown() is vulnerable to TOCTOU attacks. A call to this function should be flagged if a check function precedes it.

APIs

FunctionName	Comments
chgrp	use; deprecated or extinct in most cases
chown	use
lchown	use
fchown	
fchownat	

Method of Attack

The key issue with respect to TOCTOU vulnerabilities is that programs make assumptions about atomicity of actions. It is assumed that checking the state or identity of a targeted resource followed by an action on that resource is all one action. In reality, there is a period of time between the check and the use that allows either an attacker to intentionally or another interleaved process or thread to unintentionally change the state of the targeted resource and yield unexpected and undesired results.

The chown() call is a use-category call, which when preceded by a check-category call can be indicative of a TOCTOU vulnerability.

A TOCTOU attack in regards to chown() can occur when

a. There is a check for permissions on a file

b. Permissions on the file are changed based on the permissions state returned from the check

Between a and b, an attacker could, for example, replace the target file with a link to an attack file, resulting in an unintended file being chown'ed and, most likely, a problem with privilege escalation.

Exception Criteria

Solutions

Solution Applicability	Solution Description	Solution Efficacy
Generally applies to any chown().	Translate chown() into function(s) using file descriptors; fchown().	Effective.
Generally applies to any chown().	The most basic advice for TOCTOU vulnerabilities is to not	Does not resolve the underlying vulnerability but limits the

		perform a check before the use. This does not resolve the underlying issue of the execution of a function on a resource whose state and identity cannot be assured, but it does help to limit the false sense of security given by the check.	false sense of security given by the check.
	Generally applies to any chown().	Limit the interleaving of operations on files from multiple processes.	Does not eliminate the underlying vulnerability but can help make it more difficult to exploit.
	Generally applies to any chown().	Limit the spread of time (cycles) between the check and use of a resource.	Does not eliminate the underlying vulnerability but can help make it more difficult to exploit.
	Generally applies to any chown().	Recheck the resource after the use call to verify that the action was taken appropriately.	Effective in some cases.

Signature Details	int chown(const char *path, uid_t owner, gid_t group);
	int lchown(const char *path, uid_t owner, gid_t group);
	int fchown(int fildes, uid_t owner, gid_t group);
	int fchownat(int fildes, const char *path, uid_t owner, gid_t group, int flag);		

Examples of Incorrect Code	`/* Original UNIX V7 mkdir implementation - Scroll down to chown call */`		

```
mkdir(d)
char *d;
{
char pname[128], dname[128];
register i, slash = 0;

pname[0] = '\0';
for(i = 0; d[i]; ++i)
if(d[i] == '/')
slash = i + 1;
if(slash)
strncpy(pname, d, slash);
strcpy(pname+slash, ".");
if (access(pname, 02)) {
fprintf(stderr,"mkdir: cannot
access %s\n", pname);
++Errors;
return;
}
if ((mknod(d, 040777, 0)) < 0) {
fprintf(stderr,"mkdir: cannot
make directory %s\n", d);
++Errors;
return;
}

/* HERE IS THE TOCTOU problem 'd'
is vulnerable */
chown(d, getuid(), getgid());
strcpy(dname, d);
strcat(dname, "/.");
if((link(d, dname)) < 0) {
fprintf(stderr, "mkdir: cannot
link %s\n", dname);
unlink(d);
++Errors;
return;
}
strcat(dname, ".");
if((link(pname, dname)) < 0) {
fprintf(stderr, "mkdir: cannot
link %s\n",dname);
dname[strlen(dname)] = '\0';
unlink(dname);
unlink(d);
++Errors;
}
}
```

Examples of Corrected Code

```
struct passwd *pwd;
struct group *grp;
int fildes;
...
fildes = open("/home/cnd/mod1",
O_RDWR);
```

	`pwd = getpwnam("jones");`
	`grp = getgrnam("cnd");`
	`fchown(fildes, pwd->pw_uid, grp->gr_gid);`

Source References	• Viega, John & McGraw, Gary. Building Secure Software: How to Avoid Security Problems the Right Way. Boston, MA: Addison-Wesley Professional, 2001, ISBN: 020172152X, ch 9; • man chown() • http://www.kernelthread.com/publications/security/types.html • http://www.opengroup.org/onlinepubs/009695399/functions/fchown.html
Recommended Resource	
Discriminant Set	**Operating Systems**: • UNIX • Windows **Languages**: • C • C++

Cigital, Inc. Copyright

1. mailto:copyright@cigital.com